Active Diagnosis of Navy Machinery

Final Report

Revision 2.0

October 2016

APL
JOHNS HOPKINS
UNIVERSITY

Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel MD 20723-6099

Operating Under Contract Office of Naval Research Contract N00014-13-1-0155

Active Diagnosis of Navy Machinery

Final Report

Revision 1.0

October 2016

D. Scheidt (PI)
K. Schultz (co-PI)

Task No.: FGA63

Contract No.: N00014-13-1-0155

Contents

Introduction

Naval operations involve a complex, interconnected set of effects that are produced by the combined effects of all combat systems on platforms in the operating area. Operational effects include intelligence, surveillance and reconnaissance capabilities such as those provided by ship radar and sonar systems; defensive capabilities, such as those provided by anti-missile and anti-torpedo systems; and strike and communications systems. The combat systems that provide these effects are dependent upon electrical and fluid resources from ship auxiliary systems such as the ship's electrical distribution and potable water supply systems. Because of these dependencies, ship auxiliary system failures can cause combat load failure, which in turn, can cause mission failure.

To effectively manage a fleet of ships, commanders must understand the current and projected operating capabilities of the ships. Specifically they must understand the current and future status of the combat loads. Understanding the status of the combat loads requires that commanders understand auxiliary system status and associated impacts on combat load availability. Viewed in their entirety, the auxiliary systems of a set of ships maneuvering in a common operating area represent a massively complex, time-critical, non-linear system. The complexity of ship systems makes it impossible for human commanders to personally manage the intimate details of each auxiliary system and assess the impact individual components within the auxiliary systems have on the ship combat loads and mission success. Commanders require tools to identify mission and combat load impacts from current and predicted changes within ship auxiliary systems. For several decades now the Navy has been researching automates tools and techniques for assisting the commanders and sailors in understanding and controlling the ship auxiliary systems.

Unfortunately the highly complex, dynamic, non-linear characteristics that make ship auxiliary systems difficult for humans to manage also present challenges to machinery controls automation and, to date, no controls approach has been found that is capable of managing complex ship systems in a timely manner. The result of which is that unexpected changes in auxiliary system components can impact ship operating capabilities without informing commanders in a timely manner.

Research Goals

This effort proposed to prototype and study novel agent-based reasoning strategies and structures for ship auxiliary system diagnosis. The prototype diagnostic system addresses two Naval requirements: first, it will produce a timely understanding of the current and future auxiliary system impacts on the ship's operational status for ship commanders; second, it will produce a timely assessment of the state of key elements of the auxiliary system that can be used to produce resilient control strategies. Our investigation is founded on three key insights, which are:

1. When compared to synchronized, hierarchical agent systems, data driven, decentralized, peer-to-peer agent collaborations can significantly improve the rate, and thereby overall quality, at which diagnosis is performed. Agent collaboration strategies that allow local agents to initiate peer-to-peer diagnostic collaborations in response to local sensor data reduce the response time between an unanticipated reading that requires diagnosis and resolution. In addition, our heterogeneous strategy takes advantage of the parallel nature of distributed systems, increasing performance by producing a large number of parallel reasoning cycles, rather than the single synchronized reasoning cycle used in hierarchical systems.
2. When resilient, not optimal, control is required, system complexity can be reduced (and thereby time-to-diagnosis reduced) by ignoring those parts of the system that do not, in the current configuration, directly contribute to mission objectives.

3. Diagnosis can be improved through active diagnosis, which is a reasoning technique for improving one's knowledge by taking a control action that elicits a response that proves, or disproves, a hypothesis. For example, upon seeing a light bulb unexpectedly turn off one can distinguish a bulb failure from a power failure by taking the action of turning on a light attached to the same circuit. We hypothesize that the use of timely active diagnostic actions may reduce system complexity by eliminating large sections of the system state space as infeasible.

Agent Architecture

The prototype diagnosis system will be populated by software agents that encapsulate four reasoning components as shown in Figure 1. These components are: an intra-agent diagnostic engine that determines the current state of the agent's subsystem, a reconfiguration engine that determines control actions aimed at achieving mission objectives, an active diagnosis engine that identifies control actions aimed at improving situational awareness, and an inter-agent reasoning engine that manages exchanges between the diagnosis and reconfiguration components of outside agents. Our *research* focus lies within two of these components: inter-agent reasoning and active diagnosis.

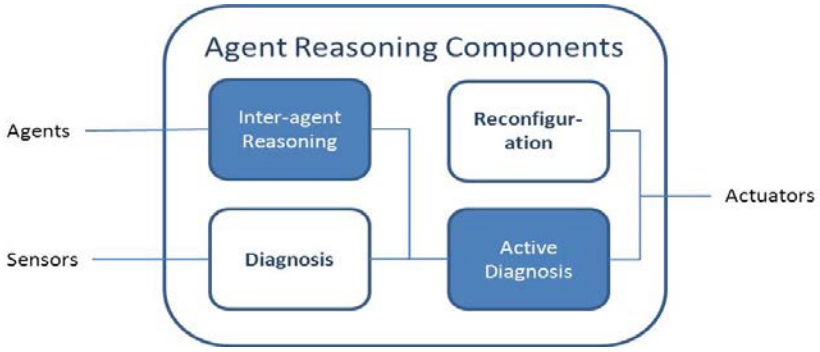


Figure 1 - Each agent has four principal reasoning components that are used to understand and act within the agent's scope and to coordinate with other agents.

To support our research we conducted experiments on the NSWC tabletop demonstrator, an experimental electrical-fluid system that exhibits qualities similar to those found on Navy ship auxiliary systems. Each component within the testbed is modeled by a hybrid, physics-based, "first principles" model that uses a finite state machine to model discrete nominal and failure states and transitions between states (e.g., Open, Closed, Stuck Open, Stuck Closed, Opening and Closing). Component behavior within a state is defined by physics-based variables (e.g., inbound flow, pressure) whose relationships are governed by propositional equations that are always true when the component is in that state or transition defined by the equations (e.g., when a valve is open inbound flow = outbound flow). Note that this approach to hybrid modeling is similar to the work of Hentzinger and Alur [8][9]. At run time, sensors and actuator values are used to assign values to variables and any propositional statement that returns false due to these values indicates a conflict between system expectations and observations.

Diagnosing failures within an agent is performed through model-based diagnosis (MBD)[10][11]. Model-based diagnosis based upon the first principles modeling technique described here has been successful at diagnosing modestly complex systems of up to 150 nodes [12][13] and diagnosing simple systems with hybrid discrete-continuous models [14].

Decomposing the system into large numbers of coupled agents with tractable spheres of responsibility allows us to frame the diagnosis as a multi-agent coordination problem, the solution to which is resolving conflicts across agent boundaries. For example, if an agent responsible for a chiller plant has

determined that either: (a) a pump has failed, which is a failure contained within the clique; or (b) the pump may not be receiving power, which is a failure outside of the clique; how does the agent determine which is the more probable solution? We propose to resolve this issue through agent-based collaborations derived from market-based resource allocation strategies. Market-based strategies that use virtual economic markets to identify optimal "price points" for resources are known to solve complex resource allocation problems [15]. Most market-based approaches rely on centralized "markets" which is counter to our objective of improving performance through decentralized, parallel processing. Our approach is similar to Sandholm's Contract Net, which uses decentralized agents that collaborate through [16] pairwise interactions to produce a global resource allocation strategy without any single agent maintaining a global model. Sandholm, and others who have used decentralized markets, used Contract Net to identify control actions. Our research is novel in that our agents will provide global diagnosis by bidding on hypotheses that satisfy local observations. Furthermore, our research focuses on using altruistic agents in the marketplace, rather than competitive, which we hope will result in faster decision convergence and configurations that are closer to optimal.

Exploiting Agent Parallelism – The proposed agent-based diagnosis system decentralizes the initiation of a diagnostic cycle which, when compared to synchronized hierarchical diagnostic systems, provides more timely responses to unexpected observations and improves processing efficiency through parallel processing. Our diagnostic strategy is based upon a resolution of conflicts between expected an actual sensor observations. Any sensor agent can initiate the diagnostic process when it determines a logical conflict between its sensor value and the expected sensor of the commanded state of the system, Allowing any sensor agent to initiate a diagnostic thread has three attractive features: (1) conflicts that can be resolved locally, and we believe that the majority of conflicts can be diagnosed within the scope of the local subsystem, which reduces complexity; (2) agents with local conflicts will operate in parallel, rather than waiting "their turn" in a universal reasoning cycle; (3) diagnostic cycles are triggered when conflicting sensor measurements are received at the local level, which accelerates the response to a conflicting observation.

Limiting Agent-Agent Dialogue – The rate of at which an agent-based system converges on a solution is non-deterministic [15] and varies as operating conditions vary. Because ship auxiliary systems are complex and dynamic, there exists a risk that the agent system will not converge on a *comprehensive* diagnosis prior to entropic forces making that diagnosis obsolete. Fortunately, auxiliary system resilience does not require comprehensive system-wide diagnosis; rather, timely diagnosis of key components that impact ship operating capabilities is required. Our agent-system achieves *resilient* diagnosis by employing an inter-agent governor that manages the agent-agent dialogue by prioritizing agent communications. Agent governors bias agent-exchanges to the most critical diagnosis assuring that the most pressing uncertainties are addressed first, even at the expense of ignoring conflicts that do not impact mission performance.

Taking Action to Understand System State – Active diagnosis is a reasoning technique that reduces uncertainty by selecting and executing a control action designed to stimulate an informative system response. Observations generated by the action allow the reasoned to identify system states as infeasible, reducing uncertainty and, at least temporarily, the system complexity. Active diagnosis has been used to aid situational awareness in a variety of problem domains including: epidemiology, agriculture, power distribution systems, and ship auxiliary systems [17]. The general strategy behind active diagnosis is to characterize the feasible states of the system being diagnosed as a set, and to identify the action that would, on average, provide information that maximizes the reduction in set size [18]. Active diagnosis systems are typically temporally and spatially passive (i.e. event driven). Motivated by human immune systems, Ishida developed an alternative approach to active diagnosis that uses

software agents to provide system-level recognition by continuously examining and mutually monitoring the host system [19]. Ishida applied this concept to a sensor network used to monitor a cement plant; each agent in Ishida's diagnostic system is associated with one of 22 sensors that monitor the cement fabrication manufacturing process. We propose to combine Ishida's insight with our insight on resilient diagnosis to create an agent-based system that continually monitors and perturbs the system to identify the state information that is most likely to be needed by Navy commanders.

Methodology

Our research effort includes a base effort and two option tasks. The base effort developed a prototype agent-based diagnosis system that investigated three principal research topics: (1) exploitation of agent parallelism by using decentralized "any-time" conflict resolution; (2) reduction of complexity by introducing a resilient agent collaboration scheme that biased diagnostic processing towards knowledge that impacts ship operating capabilities; and (3) further reduction of complexity by using active diagnosis to reduce the working state space. The base effort conducts basic and applied science aimed at understanding the behavior and efficacy of the proposed diagnostic techniques. Measurements and analysis were performed to understand the characteristics of the proposed techniques using simulation-based testing and hardware in-the-loop experimentation. The base effort used an incremental development and experimentation plan with concrete milestones and objectives spread over a three-year time frame and concluding with an integrated TRL 4 demonstration.

Experimentation was conducted using the NSWC tabletop demonstrator [20]. The demonstrator is a reduced scale variation of a ship electrical-fluid control system that includes many features found on Navy ships. A software simulation of the system developed by JHU-APL was used to perform batch testing. A reduced scale hardware system controlled by a Siemens PLC was used produce a limited set of hardware in-the-loop tests for the purposes of validating the more extensive software in-the-loop tests.

Diagnosis of Ship Auxiliary Systems

Our diagnosis algorithms are built upon a distributed agent-based framework where each agent represents only one component in the system. We restrict agents to only be able to communicate with other agents whose components are physically connected. Using model-based reasoning, the agents collectively compare their predicted behavior with the sensor values. If the predicted behavior does not match the observations then diagnosis will start and perform three main tasks: hypothesis generation, hypothesis testing, and hypothesis discrimination [21].

Passive diagnosis performs the hypothesis generation and testing and active diagnosis is used to discriminate between multiple hypotheses. Both passive and active algorithms are based upon the distributed ATMS-based consistency algorithm described in [22]. In this algorithm, each agent maintains constraints that model their hardware. From these constraints the agents generate inferences about local resources and exchange the inferences along with any assumptions that are required to make the inference. When an inference conflicts with a neighboring agent's model the assumptions and any superset of assumptions are rejected as invalid. This process is repeated until no more inferences can be made or rejected.

Component Models

To reason about the flow of fluid through its component, an agent maintains a set of internal and shared variables. The internal variables represent the component's state, the command sent to the component, and the value measured by a sensor. The shared variables represent resources that pass into and out of

a component. For the fluid system, there are two resource variables that define flow: pressure and relief. Pressure and relief are Boolean variables that are TRUE if the component has a path to a powered pump and the bucket respectively.

Each component type has a state variable and optionally a command or sensor variable. Each type also defines a custom set of constraints among its variables based on the physics of the component. Take, for example, the fluid valve shown in Figure 2. It has two neighbors A and B and has variables defining its command, state, and shared resources (See Table 1). If the valve is commanded to be open then the valve must either be open or either stuck open or stuck closed. Similarly, if it is commanded closed it can only be closed or in a stuck state. If the valve is open then the input resources will flow through the valve and generate the same value in the output variables. If the valve is closed then any input resources cannot be transferred through the component and the output resources must be FALSE. See Table 3 for a detailed summary of these constraints. We defined similar models for each of the component types: pipes, junctions, pumps, flow meters, thermal loads, check valve, and water tank. Each agent is responsible for maintaining its constraints locally and reporting updates, i.e. variable domains, to neighboring agents.

Figure 2: Schematic drawing of a valve with two neighbors: A and B.

Internal Variables		
	Command	OPEN, CLOSE
	State	OPENED, CLOSED, SO, SC
Shared Variables		
A	Pressure In	TRUE, FALSE
	Press Out	TRUE, FALSE
	Relief In	TRUE, FALSE
	Relief Out	TRUE, FALSE
B	Pressure In	TRUE, FALSE
	Press Out	TRUE, FALSE
	Relief In	TRUE, FALSE
	Relief Out	TRUE, FALSE

Table 1: The variables and their domains used to model a valve.

Pump	Command	ON, OFF
	State	ON, OFF, FAIL
Pipe	State	OK, FAIL
Flow Meter	State	TRUE, FALSE
	Sensor	FLOW, NO_FLOW
Junction	State	OK
Check Valve	State	OK

Table 2: State, command, and sensor variables for various component types.

Condition	Constraint
Command = OPEN	State ≠ CLOSED
Command = CLOSE	State ≠ OPEN
State ∈ {CLOSED, SC}	Pressure Out (A) = FALSE Pressure Out (B) = FALSE Relief Out (A) = FALSE Relief Out (B) = FALSE
State ∈ {OPEN, SO}	Pressure Out (A) = Pressure In (B) Pressure Out (B) = Pressure In (A) Relief Out (A) = Relief In (B) Relief Out (B) = Relief In (A)

Table 3: A valve's constraints on its variables

When an agent receives an update from its neighbor, it stores the intersection of its own resource variable's domain with its neighbor's domain. It then reapplies its constraints, which may cause an update to the variables. The new update is then sent to its neighbors and continues propagating throughout the system until no more updates can be made. This distributed constraint system is the foundation for both the passive and active diagnosis algorithms. In each case, the algorithm makes some assumption about one of more variables for each component and iteratively applies the constraints until all updates have finished. The algorithms differ only in which initial assumptions they make and how they interpret the results once complete.

Passive Diagnosis of Ship Auxiliary Systems

Passive diagnosis consists of two steps: hypothesis generation and hypothesis testing. Our initial implementation uses an exhaustive search to generate hypotheses. Each component will suggest any of its own failure states that satisfy the conflicts as a diagnosis. When a diagnosis is rejected each component will add its failure state to the rejected diagnosis and test it as a new diagnosis. This is not done, however, when a diagnosis is accepted because we only want the minimal diagnosis required to fully explain the conflict.

Once a hypothesis is generated the agents begin testing its validity. In our distributed constraint framework, this means that we assign the currently commanded value to the *Command* variable and the current measurement to the *Sensor* variable. Each component then assigns its *State* variable according to the diagnosis and begins propagating the results. If the diagnosis is valid then the constraints of all the components will be satisfied. However, if the diagnosis cannot result in the current measurement then there will be at least one component whose constraints cannot be satisfied.

Active Diagnosis of Ship Auxiliary Systems

Depending on how well we can observe the system state, passive diagnosis may return more than one valid diagnosis. When this occurs we need to use active diagnosis for hypothesis discrimination. Active diagnosis discovers control actions that could provide information that can be used to reject some of the hypotheses. Using the distributed constraint framework we can determine which, if any, control actions can provide additional information. To do so, we assign the *Command* and *State* variables based on a candidate action and diagnosis pair. The agent then applies the constraints and propagates the updates. When the updates have finished a *Sensor* may have a single value in its domain. If it does then this is the value that is expected at the sensor.

In some situations, the *Sensor* variable may still have both TRUE and FALSE in its domain. This occurs whenever the flow meter is part of a loop of pipe that does not contain a pump or valve. The loop creates a circular dependency that does not allow the *Sensor* values to resolve. In this case, however, we can assume that the value of the sensor will be FALSE because the variables will always resolve to only TRUE if flow is possible. Once all measurements have resolved the agents record the measured values for each control action / diagnosis pair. If a candidate action results in a different observation for two or more diagnoses then taking the action will provide new information and is kept as a candidate action. When all actions have been evaluated, the action that has the highest expected information gain is chosen and performed. Once the system stabilizes the flow meters reject any diagnoses whose expected measurement is different than the current measurement. This process is then repeated until no more candidate control actions remain that can further reduce the diagnoses.

Example Active/Passive Diagnosis Cycle

Figure 3 shows a simplified example of our fluid system. There is a single pump connected to a flow meter (FM3). Junction (J1) connects the flow meter to two branches each containing a valve (V1 and V2) and flow meter (FM1 and FM2). The branches join at a second junction (J2) before returning to the water tank. Initially, we issue the following commands:

Command 0: {P1=ON, V1=OPEN, V2=OPEN}

In this case, water should flow from the pump through the first flow meter and through both branches. We therefore expect that all three flow meters measure flow with the observation:

Observation 0: {FM1=T, FM2=T, FM3=T}

However, let's say that V1 is stuck closed and FM1 measures no flow. Because FM1 has a different value than expected it will start passive diagnosis. However, before it can start diagnosis it must create a spanning tree to ensure proper termination. Each sensor that detected a fault will suggest itself as the root and start building the spanning tree as described in the previous section. In this scenario, FM1 is the only sensor that has detected a fault and will become the root of the tree.

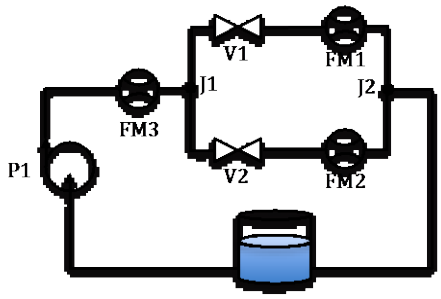


Figure 3: Simplified example of a fluid system.

FM1 will then propagate a message to all other nodes to start passive diagnosis. When a component receives the message to start diagnosis it will examine its state variable and suggests its failure state(s) as possible diagnoses. In this example, FM1 will generate the diagnosis:

Diagnosis 1: {FM1=FAIL}

It sends this as a possible diagnosis to V1 and J2 who then use their own constraints to either validate or reject the diagnosis as described earlier. FM1=FAIL is a valid diagnosis so no conflicts arise during this process. When P1 receives the message to start passive diagnosis it generates:

Diagnosis 2: {P1=FAIL_OFF}

When P1 applies its constraints its *pressure out* variable must equal FALSE. It sends this value to FM1 along with its diagnosis. However, the *pressure in* variable being FALSE conflicts with FM1's sensor variable and Diagnosis 2 is rejected. Passive diagnosis continues until the root node (FM1) is notified that all of its children have finished generating and testing their diagnoses. At the end of passive diagnosis all diagnoses are rejected except for:

Final Diagnoses: {FM1=FAIL}, {V1=SC}

Because passive diagnosis finished with multiple hypotheses we start active diagnosis. During active diagnosis each controllable component suggests its own control action as a potential action. V1 for example was commanded to be open so it suggests that it could close. Each diagnosis is examined and the measured values are determined to be:

Command 1: {V1=CLOSE}:

$\{FM1=FAIL\} \Rightarrow \{FM1=F, FM2=T, FM3=T\}$

$\{V1=SC\} \Rightarrow \{FM1=F, FM2=T, FM3=T\}$

Each diagnosis produces the same measurements and provides no new information so Command 1 is rejected. V2 also suggests that it can be closed. Each diagnosis is examined with this control action and the resulting measurements are:

Command 2: {V2=CLOSE}:

$\{FM1=FAIL\} \Rightarrow \{FM1=F, FM2=F, FM3=T\}$

$\{V1=SC\} \Rightarrow \{FM1=F, FM2=F, FM3=F\}$

Since closing V2 would result in each diagnosis producing a different measurement at FM3, Command 2 is kept as a valid action. The pump follows the same procedure suggesting that it can be turned off. This action is rejected because it would cause all flow meters to not read flow regardless of the diagnosis. At the end of the action discovery we find that there is one useful action: Command 2. V2 will then close according to this command, which results in no change in the measured values. The only diagnosis that supports the measurement is {V1=SC} so {FM1=FAIL} is rejected. Thus the combination of active and passive diagnosis generates the correct diagnosis.

Alternative Active Diagnosis Strategies

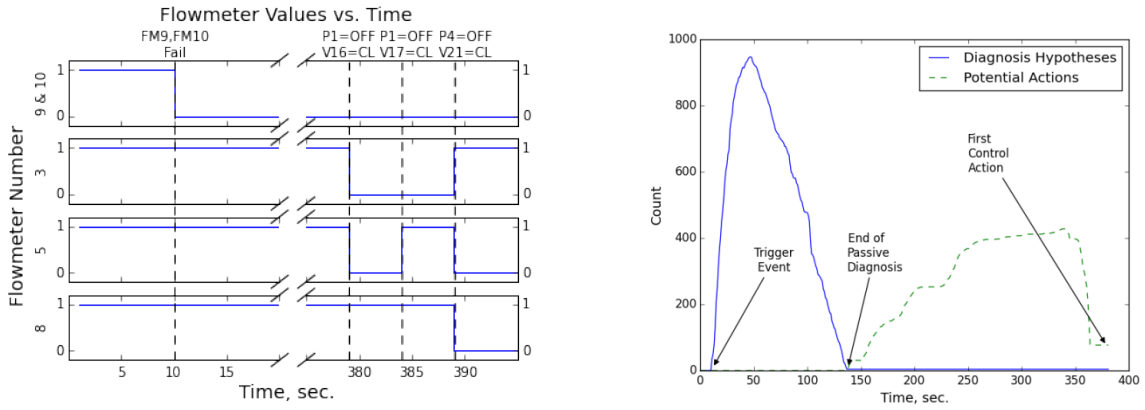
The diagnosis cycle described above is such that passive diagnosis is first performed to completion, generating the full set of feasible fault scenarios, which is then followed by the active diagnosis cycle that searches over candidate control actions to select the most informative sequence of control actions, resulting in the fewest number of control actions taken. We developed two variations on the diagnosis cycle that we hypothesized would allow for faster diagnosis and improved scalability, with a potential tradeoff of an increased number of control actions to fully diagnose the system.

The first variation proceeds as the original did by performing a passive diagnosis stage that reduces the potential fault set to the minimum size possible without changing the system state. The active stage was modified to perform control actions as soon as an informative control action was discovered, rather than performing a search over all informative control actions to determine the optimally informative sequence. The reasoning behind this modification is that the search phase for active diagnosis grows exponentially with the number of state changes allowed per control action, as well as the number of valves in the fluid system. By executing informative control actions as they are found, subsequent control actions should be found quicker, and by not exhaustively searching over the entire control space before taking control actions, the overall diagnostic process should be much faster. The tradeoff, of course, is the increased number of control actions, on average, required to fully diagnosis the fault. The second variation essentially performs passive and active diagnosis in parallel, and seeks to speed up the diagnosis process even more. In this case, the passive diagnosis starts and once it determines more than one candidate diagnostic hypothesis, the active diagnosis process starts alongside and looks for control actions that can reduce the candidate fault set and takes them, even while the passive diagnosis process is still ongoing. This requires a level of synchronization between the active and passive diagnosis processes, but should allow for an approach that scales much better (in terms of run time), as the size of the system being diagnosed increases. Both of these variations are currently undergoing batch testing to get a statistical sense of the expected performance gains over the base passive/active diagnosis approach.

Results

The effects of component failure are highly dependent on the pre-failure state of the auxiliary system; therefore it is necessary to test the diagnosis algorithms using a number of different configurations. However, the set of initial states for the tabletop system considered here is intractably large so we perform tests on only a subset of the possible initial configurations, denoted A, B, and C. Configuration A is highly redundant, with the tabletop's water flow divided roughly in half, with two pumps jointly supplying three loads. Configuration B is much more vulnerable to single points of failure, with a single pump driving all six of the loads. Configuration C has only two loads active, each driven by a single pump and isolated from each other on opposite sides of the tabletop. This configuration allows for testing when unexpected flow shows up e.g., through a pump turning on, as compared with the other two configurations, which focus on flow unexpectedly stopping.

First, consider results from a single case that demonstrates quite clearly the different stages in the diagnosis process. For this case, the system is in configuration A initially, and flowmeters 9 and 10 fail to zero at the ten second mark of the experiment. The diagnosis process is run using $N_{fail} = 2$. Figure 4a shows the values of relevant flowmeters during the diagnosis process (flowmeters not shown did not change during the course of this run). In configuration A, all flowmeters register flow initially prior to flowmeters 9 and 10 failing to zero. After passive diagnosis is completed, the algorithm then searches for informative control actions to execute. Once informative actions are determined the algorithm executes them in sequence and uses flowmeter values to determine the diagnosis. The effects of the control actions are seen in flowmeters 3, 5, and 8 which are toggled on and off by the control actions. The algorithm terminates at the 395 second mark having correctly determined the failures on flowmeters 9 and 10.



(a) (b)

Figure 4: (a) Flowmeter values during a test case (note break in time axis). (b) Counts of diagnosis hypothesis and potential control actions over time.

Figure 4b shows the number of diagnosis hypotheses and control actions being considered by the diagnosis algorithm for the same scenario as above. At the ten second mark, the change in flowmeters 9 and 10 trigger the diagnosis process which then begins generating diagnoses using passive diagnosis. While it appears that the process rapidly generates a large number of diagnoses and then winnows the set down using conflicts, the generation and rejection processes occur essentially simultaneously. However, since a hypothesis generated by one component can only be rejected by another, there is some delay in the propagation of these conflicts that reject hypotheses. Once passive diagnosis reduces the diagnosis set to the extent that it is able, the active diagnosis process generates potential control actions and rejects uninformative actions. Once the active diagnosis process determines an exhaustive list of actions, it ranks them and then proceeds to execute and evaluate candidate actions until the smallest possible diagnosis set is reached. In this particular scenario, the diagnosis set contains a single, correct diagnosis.

Batch Testing

For batch testing, the following failure states are considered: pumps fail off, valves fail open or closed, and flowmeters fail to zero. For each of the three configurations, all single component failures from the above list are tested, and N_{fail} is set to one. Many of these failures do not result in observable changes in the flowmeters, and many of these scenarios produce identical output on the flowmeters. Table 4 shows a summary of the number of observable scenarios and the total number of diagnosis hypothesis generated by passive and active diagnosis for each of the three configurations using the exhaustive single-fault set.

Configuration	# Obs. Scenarios	Total Passive	Total Active
A	24	60	36
B	21	57	35
C	20	56	24

Table 4: Summary of single-fault scenarios.

In addition to the exhaustive single component failure scenarios, a limited number of hand-selected scenarios are considered in which multiple components failed. For configuration A, pairs of flowmeters are failed. For configurations B and C, the faults only contained valve faults, and only a few fault cases

are considered. The faults for configurations B and C contain both cases where either valve failing separately do not result in a change in the flowmeters (i.e., cases not covered in the exhaustive single fault cases), and also cases that are observable individually (i.e., combinations of observable single fault cases). These results are summarized in Table 5.

Configuration	Scenarios	Total Passive	Total Active
A	35	204	35
B	3	28	5
C	7	30	8

Table 5: Summary of double-fault scenarios.

Results show that far more diagnosis hypotheses are generated in multi-fault scenarios than in the single fault cases. This increase in candidate hypotheses as failure complexity grows is a concern as complex failure scenarios on a ship may cause unmanageably large numbers of hypotheses. However, results also indicate that the active diagnosis process is able to diagnose faults more effectively in double-fault scenario cases, in terms of the ratio between total active diagnosis hypotheses to fault scenarios. This could be an artifact of the test cases chosen, however, since the algorithm parameter N_{fail} is also used to determine the maximum number of components that can change state in an active diagnosis control action the control actions in the multi-fault cases have more degrees of freedom to find effective control actions as compared with the single fault cases.

Hardware Testing

Experiments were performed using the tabletop hardware as well. However, some changes had to be made to the software infrastructure in order to support this. First, we modified the flowmeter agents to convert the multi-valued flowmeter outputs into Boolean flow values. This is done through thresholding so that the noisy readings do not constantly trigger value changes and hamper the diagnosis process. Second, the tabletop hardware expects to receive control actions for all controllable components in a single message, so we developed a middleware layer to interface the agent control actions with the tabletop PLC. This introduces some centralization into the system, but would not be required in a system where the individual agents have full control over their own state.

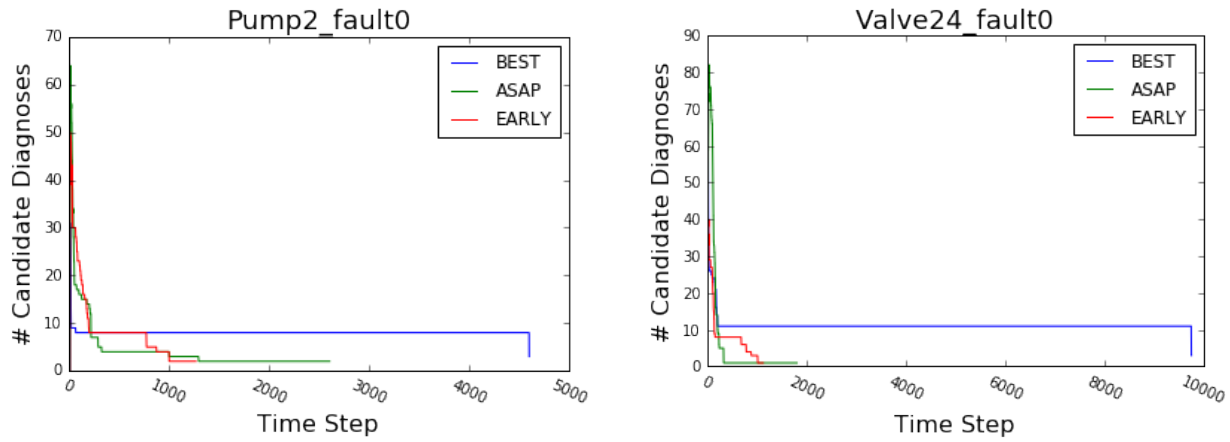
After these changes, the passive diagnosis process works in an identical manner as the low-fidelity simulation, with some caveats. In the tabletop hardware, if a pump is running but does not have pressure relief to the bucket, e.g. due to a valve failing shut or a control action, the pressure buildup generally causes a pipe to disconnect from a junction, causing water to leak. This limits the faults that are testable, since many of the faults would induce this condition. Furthermore, it makes active diagnosis risky to test, since the control actions taken may cause pressure build-ups and leaks. Despite this, we are able to successfully perform active diagnosis by carefully constructing cases where the diagnosis algorithm would not take such actions. As an additional preventative measure, we were able to construct supplementary logic in the active diagnosis control selection that would not select actions that result in pressure build up.

Alternative Diagnosis Cycles

As described above, the initial investigations into the efficacy of the passive and active diagnosis algorithms were based on a cycle where passive diagnosis was first run to completion, generating a full set of candidate diagnoses upon which the active diagnosis process performs a search over possible

informative control actions to determine the minimum number of control actions to maximally diagnose the fault. We denote this strategy ACTIVE_BEST, as it determines the most informative action set. We performed preliminary investigations in to two other strategies as well. The first, denoted ACTIVE_EARLY, performs the complete passive diagnosis process, but then takes the first informative control actions it can find, and continues to search for informative control actions until it has maximally diagnosed the fault. The second alternative, denoted ACTIVE_ASAP, essentially performs passive and active diagnosis simultaneously.

Figure 5 shows the number of candidate diagnoses over time for two different fault cases. These cases appear to be fairly typical, at least qualitatively, for the test cases we considered. As we hypothesized, the two alternative algorithms result in much quicker termination of the diagnosis process, as they do not perform an exhaustive search to evaluate every possible control action. An interesting phenomenon that we observed is that while the ACTIVE_ASAP approach generally results in fewer candidate diagnoses over the course of time, the ASAP_EARLY approach tends to actually finish first, despite generally having more active candidate diagnoses at most points in time prior to diagnosis termination.

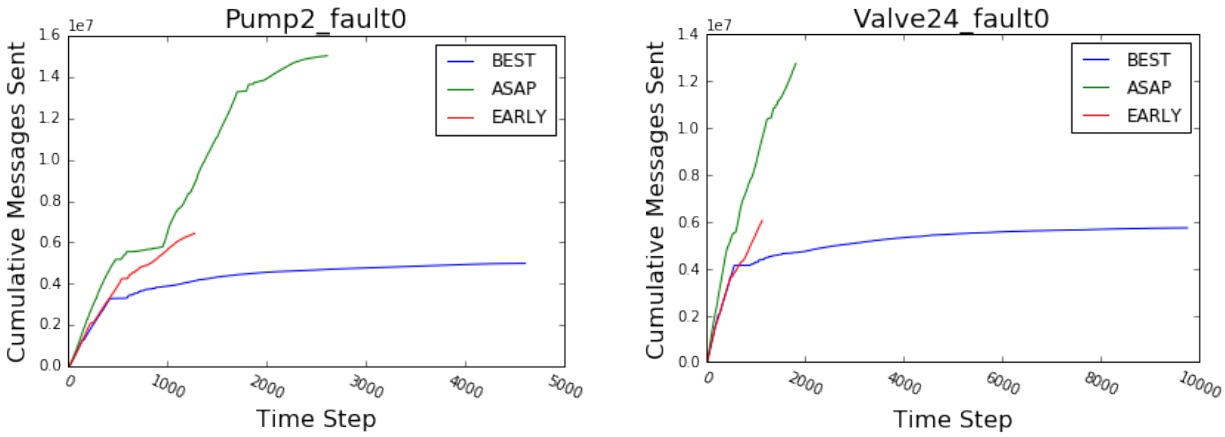


(a) (b)

Figure 5: Count of candidate diagnoses over time for two different fault scenarios.

These increases in performance come at a cost, however. Both alternative approaches use substantially more communication than the ACTIVE_BEST approach. Figure 6 shows the cumulative number of messages sent over time for different diagnosis scenarios. This is the running total of all messages sent between all agents of any type. This figure also clearly shows the run times of the different diagnostic approaches as we stop plotting at the termination point. Since ACTIVE_BEST and ACTIVE_EARLY differ only in the active diagnosis cycle and both perform the same passive diagnosis process, the amount of communication that they require track much more closely than the ACTIVE_ASAP algorithm which immediately begins generating messages for potential active diagnosis control actions once the diagnosis process starts. Due to this, there is a lot of unneeded communication that is generated, only to be overtaken by events as the passive diagnosis process winnows down the candidate diagnosis set. Additionally, from a user perspective, splitting the diagnosis process cleanly in to two stages is easier to

understand and interpret, so we conclude that the ASAP_EARLY variant is likely to be the most useful for future implementation on a naval ship.



(a) (b)

Figure 6: Cumulative count of messages sent over time for two different fault scenarios.

Summary

In conclusion, we have demonstrated a distributed, agent-based framework that is capable of diagnosing complex fault states that have multiple candidate hypotheses. Due to the designed redundancy of a ship fluid system, there exist degrees of freedom in the configuration space that allow for informative control actions to be taken to further reduce the candidate fault states.

Additional Accomplishments

The software infrastructure also underwent heavy revision and expansion during this contract. Coming in to this contract, the simulation environment consisted primarily of Matlab code and a Matlab graphical interface that communicated to the tabletop PLC. The primary simulation model used was a low fidelity Boolean flow simulation, based on a graph connectivity model that checked for the presence of flow between components. The Boolean flow simulation was re-written in Java and extended to also simulate continuous-valued steady-state hydraulic flow through the Darcy-Weisbach equations and a version of a Newton-Raphson solver to solve the nonlinear flow equations. The new software infrastructure is immensely more extensible than the previous Matlab simulation which was heavily tied to the structure of the tabletop. In the new software infrastructure, smaller fluid systems can be constructed for the purposes of unit testing, and much larger systems can be modeled in order to assess the scalability of the algorithms. New system designs are created using a graphical design tool developed under this effort. The design tool itself is extensible, allowing for additional component types to be easily defined and integrated into a new system. This is a feature we plan to use heavily in the companion planning effort, where new load and resource types will be developed and modelled. Another software tool developed on this effort was a visual display that is capable of showing the status of a simulated system, the diagnostic agents, and the physical tabletop hardware simultaneously. Many of these improvements were enabled by the development of a transparent middleware layer, so that from the developer and user perspective, the interface is identical for both flavors of simulation as well

as the tabletop. We plan to leverage these infrastructure improvements in future work, including the existing ONR effort to develop planning and load-balancing algorithms.

In addition to the primary research effort described above, there were a number of offshoots on this program that made it to various stages of completion. Graduate students contributed to two distinct efforts beyond the primary effort. The first was a completely different approach to the inference of system state and presented in [23]. Unlike the propositional logic approach discussed above, this approach used the concept of a graphical model and the loopy belief propagation algorithm to estimate the system state. The second effort focused on the diagnosis of cyber-faults, wherein an agent controlling a given component would not participate in the diagnosis process. In this case, a concept was developed wherein the remaining, functioning agents would continue to perform diagnosis through a virtual agent construct. Code was developed that demonstrated this capability in a limited sense, but it was not pursued further when the student left the project due to funding constraints. The final effort to which minimal progress was made was the notion of using hybrid logic to diagnose continuous faults in the fluid system, such as leaks. Indeed, much of the infrastructure development effort was expended with this as the eventual goal. The ability to simulate continuous flows is obviously required in this case, and it was also desired to have the ability to construct smaller fluid circuits for the purposes of unit testing the algorithms. Unfortunately, very little progress was made in actually extending the diagnosis algorithms to a hybrid model case, although the infrastructure exists to do so.

Published Papers Citing ONR Code 331 support

K. Schultz, R.Foust, K.Ramachandran, J. Harper, D. Scheidt. Distributed Multi-Agent Active Diagnosis for Ship Auxiliary Systems, Intelligent Ships Symposium XI, Philadelphia, PA, May, 2015.

F. Marungo, M. Armand, D. Scheidt, "A Residual Approach to Multi-agent Fault Detection and Isolation Using Asynchronous Belief Propagation for Fault-free Modelling of a Naval Vessel's Chilled Water System" Intelligent Ships Symposium XI, Philadelphia, PA, May, 2015.

K. Schultz, D. Scheidt, Agile Reconfiguration for Ship Auxiliary Systems, *ASNE Naval Engineers Journal* Dec 2014.

-- note: a publishable manuscript detailing the novel control methods and experimental results resulting our late FT16 efforts is being prepared for submission to the 2017 American Society of Naval Engineers Intelligent Ships Symposium (ISS 2017).

Invited Talks Citing ONR Code 331 support

Invited Panelist, Testing of Autonomous Systems Workshop, IEEE International Conference on Robotics and Automation, Stockholm, Sweden, May 2016.

NATO SCI-274 Workshop on Verification and Validation of Autonomous Systems – Invited Speaker and attendee, London, UK, 2015.

Marungo, F., Robertson, S., Quon, H., Rhee, J., Paisley, H., Taylor, R.H., and McNutt, T., "Creating a Data Science Platform for Developing Complication Risk Models for Personalized Treatment Planning in Radiation Oncology", 48th Hawaii International Conference on System Sciences (HICSS), 2015

Choo, V., "Market based reconfiguration of ship auxiliary systems, Food and Drug Administration Chapter of Sigma Xi award for best research project, Howard County STEM science fair 2015 (won by Vincent Choo, a high school intern supported under this effort).

Marungo, F., "A Residual Approach to Multi-agent Fault Detection and Isolation Using Asynchronous Belief Propagation for Fault-free Modelling of a Naval Vessel's Chilled Water System", Invited presentation to Computer Aided Medical Procedures (CAMP) Chair, faculty and students at Technical University Munich 21 August, 2015.

Invited keynote, *Resilient Cyber-Physical Systems*, 2013 IEEE International Symposium on Resilient Control Systems, San Francisco, CA, Aug, 2013.

Transition of Technical Products developed under Code 331 support

Contract HR0011-12-D-0001 T.O. 36, Distributed Battle Management, DARPA TTO; [this effort is leveraging the information theoretic control concepts originally developed under ONR Code 331 funding, albeit for a very different problem].

Contract N00024-13-D-6400, T.O. 0293; Contract Applied Unmanned Air Vehicles, ONR Code 35 [this effort is leveraging the information theoretic control concepts developed under ONR Code 331 funding, albeit for a very different problem].

Contract N00024-13-D-6400, T.O. 0724; Range Adversarial Planning Tool, Test Resource Management Center [this effort is leveraging the criticality-based test methods developed ONR Code 331 funding, albeit for a very different problem].

References

[1] Integrated Engineering Plant for Future Naval Combatants- Technology Assessment and Demonstration Roadmap APPENDIX: Contributing Team Documents MSD- 50-TR-2003/01 January 2003.

[2] Scheidt et al, Autonomous Distributed Intelligent Agents Using Model Based Reasoning, Proceedings of the 13th Ship Control Systems Symposium, Orlando, FL, USA (2003).

[3] Y. Lu, ASNE Automation & Control Symposium, ReCOVR Demonstrator, Biloxi, MS, Dec. 2007

[4] Francisco P. Maturana, Pavel Tichý, Petr Slechta, Raymond J. Staron, Fred Discenzo, Kenwood H. Hall: A Highly Distributed Intelligent Multi-agent Architecture for Industrial Automation. CEEMAS 2003: 522-532

[5] S.K. Srivastava, M. Pekala, D. Scheidt, et al.; A Control System Test Bed for Demonstration of Distributed Computational Intelligence Applied to Reconfiguring Heterogeneous Systems, IEEE Systems Conference, Honolulu, Hawaii, April, 9, 2007. [B6] D. Scheidt, K. Schultz, On Optimizing Command and Control, International Command and Control Research Technology Symposium (ICCRTS), Quebec City, ON, June 21-23, 2011.

[7] K. Drew, D. Scheidt, K. Lively, Mission-Based Engineering Plant Control, ASNE Reconfiguration and Survivability Symposium, Atlantic Beach, FL, Feb 15-18, 2005.

[8] Thomas A. Henzinger. The theory of hybrid automata. Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1996, pp. 278-292. An extended version appeared in Verification of Digital and Hybrid Systems (M.K. Inan, R.P. Kurshan, eds.), NATO ASI Series F: Computer and Systems Sciences, Vol. 170, Springer-Verlag, 2000, pp. 265-292.

[9] R. Alur, C. Coucoubetis, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. Theoretical Computer Science 138:3-34, 1995 (preliminary version appeared in Proc. Conf. on Analysis and Optimization of Systems: Discrete-event Systems, LNCIS 199, 1994).

[10] R. Reiter. A theory of diagnosis from ¯rst principles. Artif. Intell., 32:57{95}, 1987.

[11] de Kleer J. and Williams B.C.: Diagnosing multiple faults, Artificial Intelligence 32, pp.97-130, (1987).

[12] Muscettola N., P. P. Nayak, B. Pell, and B. C. Williams, "Remote Agent: To boldly go where no AI system has gone before," *Artificial Intelligence*, vol. 103, nos. 1–2, pp. 5–48, Aug. 1998.

[13] D. Alger, C. McCubbin, M. Pekala, D. Scheidt, S. Vick. Intelligent Control of Auxiliary Ship Systems. Proc. of Innovative Applications in Artificial Intelligence, AAAI, Edmonton, CA, 2002.

[14] Michael Hofbaur and Brian C. Williams. "Hybrid Estimation of Complex Systems." IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 2004.

[15] Maria Karlsson, Fredrik Ygge, Arne Andersson: Market-Based Approaches to Optimization. Computational Intelligence 23(1): 92-109 (2007)

[16] Tuomas Sandholm, "An implementation of the contract net protocol based on marginal cost calculations", In Proc. of the 11th National Conference on Artificial Intelligence (AAAI 93), July, 1993.

[17] Gregory M. Provan, Yi-Liang Chen: Agent-Based, Distributed Diagnosis for Shipboard Systems. BASYS 2002: 281-288

[18] Shakeri, M. (1996). Advances in System Fault Modeling and Diagnosis. Ph.D. thesis, University of Connecticut.

[19] Yoshiteru Ishida (1997) Active Diagnosis by Self-Organization : An Approach by The Immune Network Metaphor In: International Joint Conference on Artificial Intelligence 1084-1089

[20] Srivastava, S.K.; Cartes, D.A.; Maturana, F.; Ferrese, F.; Pekala, M.; Zink, M.; Meeker, R.; Carnahan, D.; Staron, R.; Scheidt, D.; Huang, K**.** A Control System Test Bed for Demonstration of Distributed Computational Intelligence Applied to Reconfiguring Heterogeneous Systems, *IEEE Instrumentation and Measurement Magazine*, Vol 11: Issue 1, Page(s): 30-37 Feb 2008.

[21] Davis, R., & Hamscher, W. (1988). Model Based Trouble Shooting. In H. Shrobe (gds.), Exploring Artificial Intelligence Morgan Kaufmann .

[22] Makoto Yokoo, Katsutoshi Hirayama: Algorithms for Distributed Constraint Satisfaction: A Review. Autonomous Agents and Multi-Agent Systems 3(2): 185-207 (2000)

[23] F. Marungo, M. Armand, D. Scheidt, "A Residual Approach to Multi-agent Fault Detection and Isolation Using Asynchronous Belief Propagation for Fault-free Modelling of a Naval Vessel's Chilled Water System" Intelligent Ships Symposium XI, Philadelphia, PA, May, 2015.